REMARKS

The Office action has been carefully considered. The Office action has maintained all previous claim rejections. Specifically, the Office action rejected claims 32-41 under U.S.C. §101 as being directed to non-statutory subject matter. Further, the Office action rejected claims 1-17, 19-22, 25-31, 42-43, 45-47, and 49 under U.S.C. §102(e) as being anticipated by U.S. Patent No. 5,974,470 to Hammond et al. ("Hammond"). Still further, the Office action rejected claims 32-41 under U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,185,734 to Saboff et al. ("Saboff"). Further yet, the Office action rejected claims 18, 24, and 44 under U.S.C. §103(a) as being unpatentable over Hammond. Finally, the Office action rejected claims 23 and 48 under U.S.C. §103(a) as being unpatentable over Hammond in view of Saboff.

In addition to the claim rejections noted above, the Office action has provisionally rejected claims 1, 3-5, 7-22, 24-26, 31, 42, 43 and 45-49 under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-4, 12, 16-22, 26, and 27 of copending U.S. Application No. 09/842,278. As noted in the Office action, applicants will timely file a terminal disclaimer with respect to these claims upon indication of allowable subject matter.

The Office action also objected to the specification over a typographical error. Applicants have corrected the error in the specification which eliminates any erroneous reference to an incorrect copending application, thus, obviating the objection.

By present amendment, claims 1, 16, 32, 39, 41and 42 have been amended for clarification and not in view of the prior art. Applicants submit that the claims as filed were patentable over the prior art of record, and that the amendments herein are for purposes of clarifying the claims and/or for expediting allowance of the claims and not for reasons related to patentability. Reconsideration is respectfully requested.

Applicants thank the Examiner for the interview held (by telephone) on March 8, 2005. During the interview, the Examiner and applicants' attorney discussed the claims with respect to the prior art. The essence of applicants' position is incorporated in the remarks below.

Prior to discussing reasons why applicants believe that the claims in this application are clearly allowable in view of the teachings of the cited and applied references, a brief description of the present invention is presented.

The present invention is directed to a system and method for an infrastructure that allows applications to run with specified versions of dependent assemblies, wherein each assembly may exist and run side-by-side on the system with other versions of the same assembly being used by other applications. More specifically, when executed, an application may provide a manifest to specify any desired assembly versions on which it is dependent. Similarly, each assembly may have an assembly manifest that specifies the versions of assemblies on which it is dependent. The particular versions of the assemblies required by the applications may be tracked in a separate and distinct file such that if assemblies change as updates are implemented, the application does not also need to be changed and updated, rather just the manifest. In this manner, the application that requires a particular version of an assembly may retrieve the particular version of the assembly while other applications that are not as dependent on a very specific version may use the updated version of assembles as they become available.

For example, during an initialization phase, *i.e.*, when an application is first launched, an activation context may be created for the application based on the manifests to map version independent names to a particular assembly version maintained on the system. While the application is in a running phase, for any globally named object that the application wants created, the activation context may be accessed to locate the application's or assembly's manifest-specified version. The manifests and activation context constructed therefrom, thus, may isolate an application from assembly version changes.

Note that the above description is for example and informational purposes only, and should not be used to interpret the claims, which are discussed below.

§101 Claim Rejections

The Office action rejected claims 32-41 as being directed to non-statutory subject matter because claims 32-41 are recited as data structures stored on computer-readable media and are merely stored so as to be read or outputted by a computer without creating any functional interrelationships. Applicants respectfully disagree and submit that claims 32-41 are directed toward a computer-readable medium having stored thereon a data structure. MPEP § 2106(IV)(B)(1a) specifically states that "a claimed computer readable medium encoded with a data structure *defines* (emphasis added) structural and functional interrelationships between the data structure and the medium which permit the data structure's functionality to be realized, and is thus statutory." In contrast, MPEP § 2106(IV)(B)(1a) goes on to state further "[d]ata structures *not claimed as embodied in computer-readable media* (emphasis added) are descriptive material per se and are not statutory because they are not capable of causing functional change in the computer." By organizing a data structure with the claimed sets of data on a computer readable medium, structural and functional interrelationships between the data structure and the medium are realized and defined as statutory *per se*. Applicants maintain that claims 32-41, as originally entered recite statutory subject matter. Notwithstanding, applicants have nevertheless amended claims 32, 39, and 41 to recite additional sufficient functionality with respect to the data structures so as to remove any doubt whatsoever about the statutory nature of the recitations. Applicants respectfully submit that claims 32-41 are directed to statutory subject matter and no further correction is required.

§102 Claim Rejections citing Hammond

Turning to the first independent claim, amended claim 1 recites a computer-implemented method, comprising receiving a request from executable code to load an assembly, the request not including assembly version data, building an activation context associated with the executable code in response to the request to load an assembly, consulting information in an a manifest associated with the executable code and stored separate from the executable code to determine a particular version of the assembly in response to the building of the activation context, and providing the particular version of the assembly for use by the executable code.

The Office action rejected claim 1 as being anticipated by Hammond. More specifically, the Office action contends that Hammond teaches receiving a request from executable code to load an assembly, the request not including assembly version data. Column 5, line 58 to column 6, line 12 of Hammond is referenced. Further, the Office action contends that Hammond teaches consulting information associated with the executable code to determine a particular version of the assembly. Column 7, line 51 to column 8, line 5 of Hammond is referenced. Finally, the Office action contends that Hammond teaches providing the particular version of the assembly for use by the executable code. Column 5, lines 27-30 of Hammond is referenced. Applicants respectfully disagree.

As has been presented in the previous Office action response, Hammond is directed, generally, to a software method of providing a patch to an operating system that modifies the Application Program Interface (API) call logic. When a program is executed, any called modules are then called according to the new call logic instead of the original call logic of the operating system. See abstract, generally. The software patch provides the user with the option of setting "location rules" that govern what path that the call logic will use to locate the called modules. See column 5, lines 34-40 of Hammond.

One such location rule (specifically cited by the Office action) that may be set is the "version rule." When set, the version rule bit is an indication to the call logic that only one specific module must be called in conjunction with the application. However, "version" is simply a name given to the rule to distinguish between calling any module and calling a specific module. The actual version, however, is not specified by the system disclosed in Hammond. Rather, when the version rule is set, the call logic will only look to one location (typically the same directory as the application itself) to find the module, but the call logic remains unaware of the actual version of the module being called. See column 7, lines 51-67 of Hammond. Hammond refers to an example, wherein two DLL files are referred to as version 1.0 and version 2.0, but the call logic is simply unaware of the DLL file being called. The version rule merely directs the call logic to a location which allows the call logic to load whatever DLL file happens to be in the directory, regardless of what it may be named, (*i.e.*, it will load version 2.0 because it is located at a specified location and not because it is the second version of the DLL). Thus, despite the confusing choice of Hammond to refer to the location-determining bit as the "version rule," Hammond does not teach distinguishing between actual versions of DLL files or any other files, modules or assemblies.

In contrast, claim 1 recites building an activation context associated with the executable code in response to a request to load an assembly and consulting information in a manifest associated with the executable code and stored separate from the executable code to determine a particular version of the assembly in response to the building of the activation context. The method of the present invention may consult information stored in an activation context to determine which actual version of an assembly may be needed by the executable code.

Determining the actual version of an assembly is not the same as determining the location of a DLL file that happens to be named by a version number.

Furthermore, claim 1 recites the request not including assembly version data. Clearly, the system in Hammond must necessarily include version data in a request to load a module or the location rules would not function properly. Hammond admits as much by providing an example of a situation where ten applications require "version 1.0" of a particular module and one other application requires "version 2.0" Obviously, the request to load these module must necessarily include the versioning information in order for the system of Hammond to apply the "search rule," the "alias rule," and the "version rule."

Notwithstanding these differences, claim 1 has been amended to recite building an activation context associated with the executable code in response to the request to load an assembly. Certainly, Hammond cannot be construed to teach this recitation as Hammond does not teach or is even cognizant of the concept of building an activation context in response to a request to load an assembly.

Thus, the teachings of Hammond do not disclose the method recited in claim 1. For at least the foregoing reasons, applicants submit that claim 1 is allowable over the prior art of record.

Applicants respectfully submit that dependent claims 2-15, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 1 and consequently includes the recitations of independent claim 1. As discussed above, Hammond fails to disclose the recitations of claim 1 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 1 noted above, each of these dependent claims includes additional patentable elements.

For example, claim 5 recites that consulting information associated with the executable code to determine a particular version of the assembly includes searching for a mapping from a version independent name provided by the executable code to a version specific assembly. As discussed above, Hammond does not teach consulting information associated with the executable code to determine a particular version of the assembly. Specifically, Hammond teaches determining the location of a DLL file specified to be loaded and cannot distinguish between versions. Therefore, Hammond cannot possibly teach searching for a mapping from a version independent name provided by the executable code to a version specific assembly. Although, Hammond discloses a "version map," this concept in Hammond is merely an identification of aliases used when multiple calls of identically named modules are made. Hammond provides a means for renaming modules that happen to be named by the same file name (library.dll, for example) that are called more than once and a means for keeping track of the resulting names in what Hammond has chosen to call "a version map." The version map of Hammond, which is not actually aware of versioning information of its stored files, cannot be construed to read on identifying specific versions of an assembly that are mapped to version independent names of assemblies as recited in claim 5. Applicants submit that claim 5 is allowable for at least this additional reason.

Turning to the next independent claim, claim 16 recites a computer-implemented method, comprising building an activation context associated with executable code, the activation context identifying dependency information, interpreting the dependency information associated with the executable code, the dependency information identifying at least one particular version of an assembly, and associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information.

The Office action rejected claim 16 as being anticipated by Hammond. More specifically, the Office action contends that Hammond teaches interpreting dependency information associated with executable code, the dependency information identifying at least one particular version of an assembly. Column 7, line 51 to column 8, line 5 of Hammond is referenced. Further, the Office action contends that Hammond teaches associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information. Again, Column 7, line 51 to column 8, line 5 of Hammond is referenced. Applicants respectfully disagree.

As discussed above, the method and system disclosed in Hammond does not teach utilizing information associated with the application itself to determine which actual version of a module is required during execution. Rather, Hammond teaches a system and method for providing location rules that assist the operating system in determining the location of modules to load when applications are executed. Hammond does not teach interpreting dependency information associated with executable code, the dependency information identifying at least one particular version of an assembly, as recited in claim 16. At best, Hammond teaches interpreting location rules to determine the location of the module to call.

Furthermore, since Hammond only teaches determining the location of modules to call, Hammond cannot possibly be construed to teach associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information as recited in claim 16. Again, at best, Hammond teaches mapping different alias names assigned to multiples calls of the same version of a single module.

Notwithstanding these differences, claim 16 has been amended to recite building an activation context associated with executable code, the activation context identifying dependency information. Certainly, Hammond cannot be construed to teach this recitation as Hammond does not teach or is even cognizant of the concept of an activation context identifying dependency information.

For at least the foregoing reasons, applicants submit that claim 16 is allowable over the prior art of record for at least these reasons.

Applicants respectfully submit that dependent claims 17, 19-22 and 25-31 by similar analysis are allowable. Each of these claims depends either directly or indirectly from claim 16 and consequently includes the recitations of independent claim 16. As discussed above, Hammond fails to disclose the recitations of claim 16 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 16 noted above, each of these dependent claims includes additional patentable elements.

Turning to the next independent claim alleged as anticipated by Hammond, claim 42 recites a system in a computing environment, comprising an initialization mechanism configured to interpret dependency data associated with executable code, the dependency data corresponding to at least one assembly version on which the executable code depends, an activation context, the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data, the activation context relating at least one version independent assembly identifier provided by the executable code to a version specific assembly, and a version matching mechanism configured to access the activation context to relate a version independent request from the executable code to a version specific assembly.

The Office action specifically contends that Hammond teaches an initialization mechanism configured to interpret dependency data associated with executable code, the dependency data corresponding to at least one assembly version on which the executable code depends. Column 7, line 51 to column 8, line 5 of Hammond is referenced. The Office action also contends that Hammond teaches an activation context, the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data, the activation context relating at least one version independent assembly identifier to a version specific assembly. Column 8, lines 23-58 of Hammond is referenced. Finally, the Office action contends that Hammond teaches a version matching mechanism configured to access the activation context to relate a version independent request from the executable code to a version specific assembly. Column 7, line 51 to column 8, line 5 of Hammond is referenced. Applicants respectfully disagree.

As discussed above, the method and system disclosed in Hammond does not teach utilizing information associated with the application itself to determine which actual version of a module is required during execution. Rather, Hammond teaches a system and method for providing location rules that assist the operating system in merely determining the location of modules to load when applications are executed. In terms of claim 42, Hammond does not teach an initialization mechanism configured to interpret dependency data associated with executable code, the dependency data corresponding to at least one assembly version on which the executable code depends. At best, Hammond teaches interpreting location rules to determine the location of the module to call.

Furthermore, no where in Hammond can there be found any teaching of an activation context. For that matter, Hammond does not show any cognizance of an activation context, let alone the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data as recited in claim 42. Again, at best, Hammond teaches a version map for mapping different alias names assigned to multiples calls of the identically named modules.

For at least the foregoing reasons, applicants submit that claim 42 is allowable over the prior art of record.

Applicants respectfully submit that dependent claims 43, 45-47, and 49 by similar analysis are allowable. Each of these claims depends either directly or indirectly from claim 42 and consequently includes the recitations of independent claim 42. As discussed above, Hammond fails to disclose the recitations of claim 42 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 42 noted above, each of these dependent claims includes additional patentable elements.

§102 Claim Rejections citing Saboff

Amended claim 32 recites a computer-readable medium having stored thereon a data structure, comprising a first data store operable to store a first set of data comprising a name of an assembly, a second data store operable to store a second set of data comprising a version of the assembly, a third data store operable to store a third set of data comprising at least one item of the assembly, and a fourth data store operable to store a fourth set of data comprising binding path data to each item in the third set of data, wherein each data store is operable to provide information to an activation context when executable code is executed.

The Office action rejected claim 32 as being anticipated by Saboff. More particularly, the Office action contends that Saboff teaches a first set of data comprising a name of an assembly. Service 509 in FIG. 6 and column 9, lines 11-18 of Saboff are referenced. Further, the Office action contends that Saboff teaches a second set of data comprising a version of the assembly. Service 513 in

FIG. 6 and column 9, lines 53-55 of Saboff are referenced. Still further, the Office action contends that Saboff teaches a third set of data comprising at least one item of the assembly. State type 510 and 511 of FIG. 6 and column 9, lines 19-39 of Saboff are referenced. Finally, the Office action contends that Saboff teaches a fourth set of data comprising binding path data to each item in the third set of data. Path 507 of FIG. 6 and column 9, lines 5-7 of Saboff are referenced. Applicants respectfully disagree.

Saboff is directed, generally, to a registry for maintaining versions of services such that multiple applications may use the various versions of the same services. Accordingly, Saboff discloses a data structure having a number of fields that contain values for use in its service maintenance system. In particular, Saboff discloses a data structure having a name of a service (509), a version of the service (513), a type or state of the service (510 and 511), and a path to where the service is located (507).

In contrast, claim 32 recites a first set of data comprising an assembly. As argued in the previous Office action response, an assembly is not a service as in Saboff. Further, claim 32 recites a third set of data comprising at least one item of the assembly. Again, an assembly is not a service and one item in the assembly is quite different from the type or state of a service. Thus, the contention by the Office action that the disclosures of Saboff correspond on a one-for-one basis to the recitations of claim 32 is not supported by Saboff.

Furthermore, Saboff cannot possibly teach the recitations of claim 32 because of the nature in which the fields of the data structure differ. Saboff merely teaches the use of path data to indicate the location of the service identified in the service field. Quite differently, claim 32 recites a fourth set of data comprising binding path data to each item in the third set of data. That is, the binding path does not represent a location of the assembly itself, but rather a relationship between each item in the third set of data and an outside source, such as an API or call program. Not only does the data in the fourth set correspond to a different set of data than what is disclosed in Saboff (item in an assembly as opposed to a service), it also corresponds in a different manner (binding path as opposed to mere location). A binding path necessarily implies a relationship between two items whereas a location merely indicates information about one item. Applicants submit that claim 32 is allowable over the prior art for at least the foregoing reasons.

Applicants respectfully submit that dependent claims 32-38, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 32 and consequently includes the recitations of independent claim 32. As discussed above, Saboff fails to disclose the recitations of claim 32 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 32 noted above, each of these dependent claims includes additional patentable elements.

Regarding independent claims 39 and 41 as well as dependent claim 40, these claims include recitations generally directed to a computer-readable medium encoded with a data structure having a set of data generally relating to an assembly. As discussed above regarding the recitation of claims 32-38, an assembly is not a service as described in Saboff, and, consequently by similar analysis, Saboff also fails to disclose the recitations of claims 39-41. Furthermore, claims 39-41 are allowable for their additional patentable elements. Applicants submit that claims 39-41 are allowable over the prior art of record for at least these reasons.

§103(a) Claim Rejections

The Office action rejected claims 18, 23, 24, 44, and 48 as being unpatentable either over Hammond alone or in view of Saboff. Claims 18 and 23-24 depend from claim 16 and claims 44 and 48 depend from claim 42. Hammond, whether considered alone or in any permissible combination at law with any prior art of record, including Saboff, does not teach or suggest the recitations of claims 16 and 42 respectively for at least the reasons discussed above with respect to the §102 rejections of these independent claims. Thus, applicants submit that each of these dependent claims are also allowable for at least the reasons that each respective independent claim is allowable as was discussed above with respect to the §102 rejections. More particularly, applicants submit that claims 18, 23, and 24 are allowable for at least the reasons that claim 16, the claim from which these claims depend, is allowable. Additionally, applicants submit that claims 44 and 48 are allowable for at least the reasons that claim 42, the claim from which these claims depend, is allowable.

For example, neither Hammond or any other prior art of record, whether considered alone or in any permissible combination at law, discloses or suggests the recitation of an application manifest associated with the executable code by being stored in a common folder with an application executable file that corresponds to the executable code as recited in claim 18. Nor does Hammond or any other prior art of record, whether considered alone or in any permissible combination at law, disclose or suggest the recitations of an initialization mechanism adding information that corresponds to assembly dependency data to an activation context as recited in claim 48. Further, by law, in order to modify a reference to reject claimed subject matter, there must be some teaching or suggestion outside of applicants' teachings to do so. Neither Hammond nor Saboff have any such teachings or suggestions as to any such modification, let alone any teaching or suggestion as to how either Hammond's system or Saboff's system could be modified, or why it might be desirable to do so. The only other way in which Hammond or Saboff could be modified to reach applicants' claimed invention is via applicants' own teachings, which is impermissible by law.

For at least the reasons discussed above with respect to both the §102 and §103 rejections, applicants submit that all the claims are patentable over the prior art of record. Reconsideration and withdrawal of the rejections in the Office action is respectfully requested and early allowance of this application is earnestly solicited.

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that claims 1-49 are patentable over the prior art of record, and that the application is in good and proper form for allowance. A favorable action on the part of the Examiner is earnestly solicited.

If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney at (425) 836-3030.

Respectfully submitted,

Albert S. Michalik, Reg. No. 37,395
Attorney for Applicants
Law Offices of Albert S. Michalik, PLLC
704 - 228th Avenue NE, Suite 193
Sammamish, WA 98074
(425) 836-3030
(425) 836-8957 (facsimile)

In re Application of GRIER et al.
Serial No. 09/842,270

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that this Response, along with transmittal, credit card payment form, petition for extension of time, and facsimile cover sheet, are being transmitted by facsimile to the United States Patent and Trademark Office in accordance with 37 C.F.R. 1.6(d) on the date shown below:

Date: May 20, 2005

Albert S Michalik
Albert S. Michalik

2501 Second Amendment